Class15

Andres Rivero

11/18/2021

Background

Today we examine a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

Load the contData and colData

We need 2 things - 1: count data - 2: colData (the metadata that tells us about the design of the experiment).

```
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <- read.csv("airway_metadata.csv")
```

```
head(counts)
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003        723        486        904        445       1170
## ENSG00000000005          0          0          0          0          0
## ENSG00000000419        467        523        616        371        582
## ENSG00000000457        347        258        364        237        318
## ENSG00000000460         96         81         73         66        118
## ENSG00000000938          0          0          1          0          2
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003       1097        806        604
## ENSG00000000005          0          0          0
## ENSG00000000419        781        417        509
## ENSG00000000457        447        330        324
## ENSG00000000460         94        102         74
## ENSG00000000938          0          0          0
```

```
head(metadata)
```

```
##           id     dex celltype     geo_id
## 1 SRR1039508 control   N61311 GSM1275862
## 2 SRR1039509 treated   N61311 GSM1275863
## 3 SRR1039512 control  N052611 GSM1275866
## 4 SRR1039513 treated  N052611 GSM1275867
## 5 SRR1039516 control  N080611 GSM1275870
## 6 SRR1039517 treated  N080611 GSM1275871
```

Side-note: Let's check the corespondence of the metadata and count data setup.

```
metadata$id
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
colnames(counts)
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

We can use the == thing to see if they are the same

```
all(metadata$id == colnames(counts))
```

```
## [1] TRUE
```

```
all( c(T,T,T,T,T,T,F) )
```

```
## [1] FALSE
```

Compare control to treated

First wee need to access all the control columns in our counts data.

```
control.inds <- metadata$dex == "control"
control.ids <- metadata[ control.inds, ]$id
```

Use these ids to access just the control columns of our `counts` data

```
control.mean <- rowMeans(counts[ , control.ids] )
head(control.mean)
```

```
## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
##          900.75            0.00          520.50          339.75           97.25
## ENSG00000000938
##            0.75
```

Do the same for the drug treated...

```
treated.id <- metadata[ metadata$dex == "treated", ]$id
treated.mean <- rowMeans(counts[,treated.id])
```

We will combine our meancount data for bookkeeping purposes.

```
meancounts <- data.frame(control.mean, treated.mean)
```

There are 38694 rows/genes in this dataset

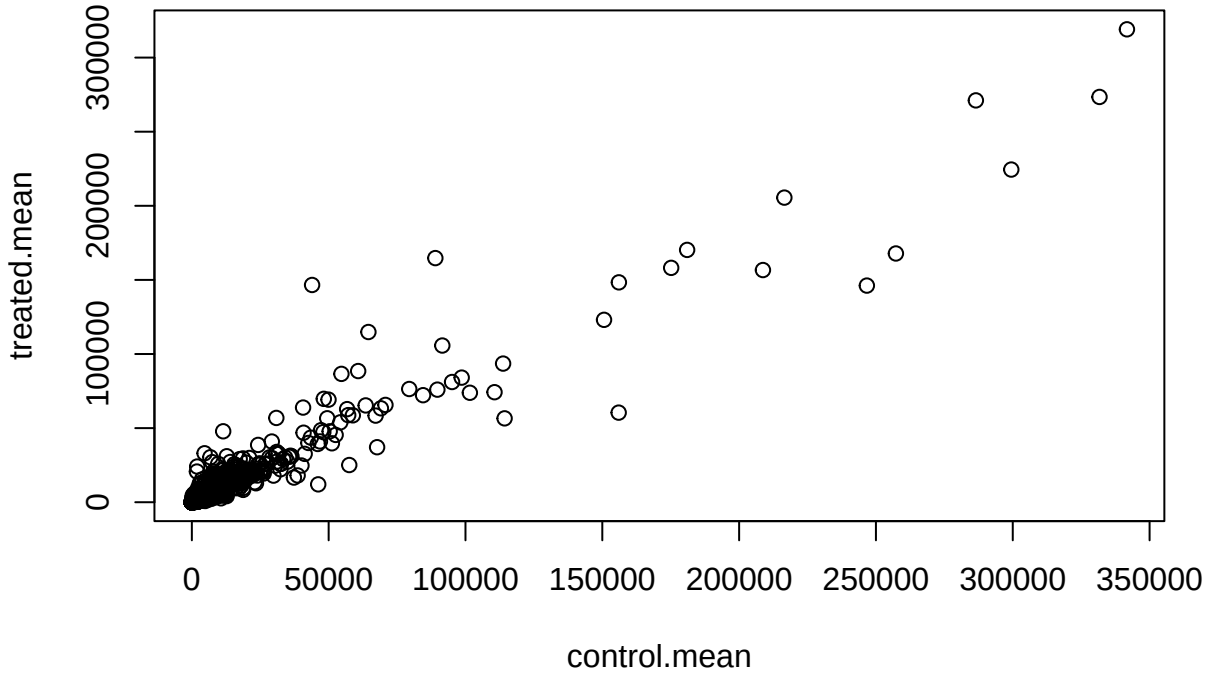
```
nrow(counts)
```

```
## [1] 38694
```

Compare the control and treated

A quick plot of our progress so far

```
plot(meancounts)
```

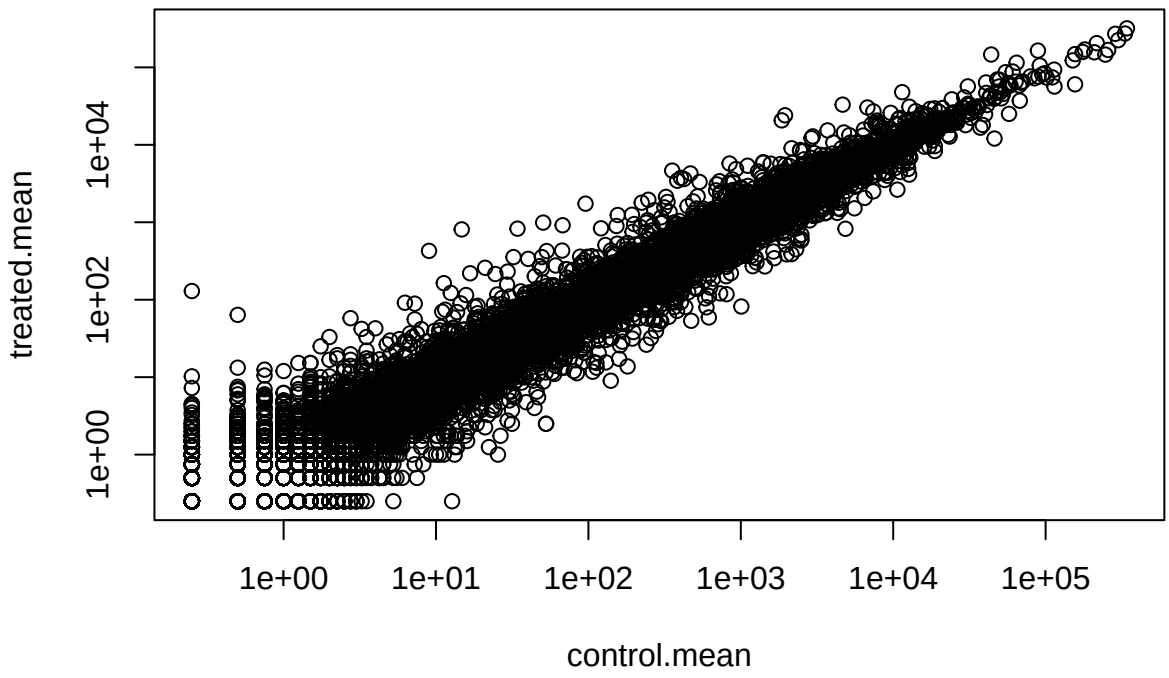


This would benefit from a log transform! Let's plot on a log log scale

```
plot(meancounts, log="xy")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted
## from logarithmic plot
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
## from logarithmic plot
```



We often use log transformations as they make life much nicer in this world...

```
log2(20/20)
```

```
## [1] 0
```

```
log2(40/20)
```

```
## [1] 1
```

```
log2(10/20)
```

```
## [1] -1
```

```
log2(80/20)
```

```
## [1] 2
```

Cool. I like log2!

```
meancounts$log2fc <- log2(meancounts[,"treated.mean"]/
                          meancounts[,"control.mean"])
head(meancounts)
```

```
##                 control.mean treated.mean      log2fc
## ENSG00000000003       900.75       658.00 -0.45303916
## ENSG00000000005         0.00         0.00         NaN
## ENSG00000000419       520.50       546.00  0.06900279
## ENSG00000000457       339.75       316.50 -0.10226805
## ENSG00000000460        97.25        78.75 -0.30441833
## ENSG00000000938         0.75         0.00        -Inf
```

We need to drop the zero count genes/rows!

```
head(meancounts[,1:2])
```

```
##                 control.mean treated.mean
## ENSG00000000003       900.75       658.00
## ENSG00000000005         0.00         0.00
## ENSG00000000419       520.50       546.00
## ENSG00000000457       339.75       316.50
## ENSG00000000460        97.25        78.75
## ENSG00000000938         0.75         0.00
```

```
head(meancounts[,1:2] == 0)
```

```
##                 control.mean treated.mean
## ENSG00000000003        FALSE        FALSE
## ENSG00000000005         TRUE         TRUE
## ENSG00000000419        FALSE        FALSE
## ENSG00000000457        FALSE        FALSE
## ENSG00000000460        FALSE        FALSE
## ENSG00000000938        FALSE         TRUE
```

The `which()` function tells us the indices of TRUE entries in a logical vector.

```
which( c(T,F,T) )
```

```
## [1] 1 3
```

However, it is not that useful in default mode on our type of multi column input. . . .

```
inds <- which(meancounts[,1:2] == 0, arr.ind=TRUE)
head(inds)
```

```
##                 row col
## ENSG00000000005   2   1
## ENSG00000004848  65   1
## ENSG00000004948  70   1
## ENSG00000005001  73   1
## ENSG00000006059 121   1
## ENSG00000006071 123   1
```

I only care about the rows here (if there is a zero in any column I will exclude this row eventually).

```
to.rm <- unique(sort(inds[,"row"]))
```

```
mycounts <- meancounts[-to.rm,]
head(mycounts )
```

```
##                 control.mean treated.mean      log2fc
## ENSG00000000003       900.75       658.00 -0.45303916
## ENSG00000000419       520.50       546.00  0.06900279
## ENSG00000000457       339.75       316.50 -0.10226805
## ENSG00000000460        97.25        78.75 -0.30441833
## ENSG00000000971      5219.00      6687.50  0.35769358
## ENSG00000001036      2327.00      1785.75 -0.38194109
```

We now have 21817 genes remaining.

```
nrow(mycounts)
```

```
## [1] 21817
```

How many of these genes are up regulated at the log2 fold-change threshold of +2 or greater?

```
sum(mycounts$log2fc > +2)
```

```
## [1] 250
```

What percentage is this?

```
round((sum(mycounts$log2fc > +2) / nrow(mycounts))*100, 2)
```

```
## [1] 1.15
```

How about down?

```
sum(mycounts < -2)
```

```
## [1] 367
```

DESeq2 analysis

```
library(DESeq2)
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

We first need to setup the DESeq input object.

```
dds <- DESeqDataSetFromMatrix(countData=counts,
                              colData=metadata,
                              design=~dex)
```

```
## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

Run the DESeq analysis pipeline.

```
dds <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

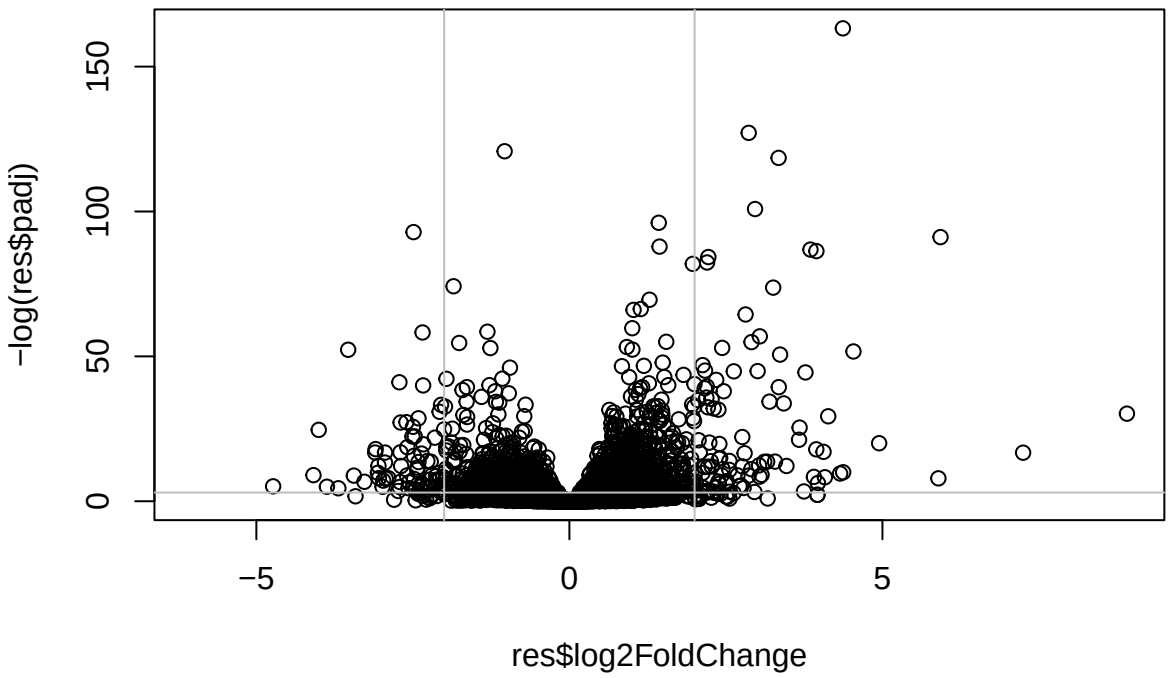
```
res <- results(dds)
head(res)
```

```
## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 6 columns
##                    baseMean log2FoldChange     lfcSE      stat    pvalue
##                   <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
## ENSG00000000005   0.000000             NA        NA        NA        NA
## ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
## ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
## ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
## ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
##                      padj
##                 <numeric>
## ENSG00000000003  0.163035
## ENSG00000000005        NA
## ENSG00000000419  0.176032
## ENSG00000000457  0.961694
## ENSG00000000460  0.815849
## ENSG00000000938        NA
```

A Volcano plot

This is a very common data viz of this type of data that does not really look like a volcano.

```
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2,2), col="gray")
abline(h=-log(0.05), col="gray" )
```



Adding annotation data

We want to add meaningful gene names to our dataset so we can make some sense of what is going on here...

For this we will use two bioconductor packages, one does the work and is called **AnnotationDbi** the other contains the data we are going to map between and is called **org.Hs.eg.db**.

First I need to install with `BiocManager::install("org.Hs.eg.db")` and `BiocManager::install("AnnotationDbi")`.

```
library(AnnotationDbi)
library(org.Hs.eg.db)
```

```
##
```

```
columns(org.Hs.eg.db)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
## [16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
## [21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [26] "UNIPROT"
```

Here we map to "SYMBOL" the comon gene name that the world understands and wants.

```
res$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(res),
                     keytype="ENSEMBL",
                     column="SYMBOL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
head(res)
```

```
## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 7 columns
##                   baseMean log2FoldChange     lfcSE      stat    pvalue
##                  <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
## ENSG00000000005   0.000000             NA        NA        NA        NA
## ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
## ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
## ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
## ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
##                      padj      symbol
##                 <numeric> <character>
## ENSG00000000003  0.163035      TSPAN6
## ENSG00000000005        NA        TNMD
## ENSG00000000419  0.176032        DPM1
## ENSG00000000457  0.961694       SCYL3
## ENSG00000000460  0.815849    C1orf112
## ENSG00000000938        NA         FGR
```